|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_Информатика и системы управления\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_Программное обеспечение ЭВМ и информационные технологии» (ИУ7)\_\_\_\_\_\_\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К КУРСОВОМУ ПРОЕКТУ***

***НА ТЕМУ:***

***Прототип интернет-магазина видеоигр \_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Студент **\_\_\_\_ИУ7-21М\_\_\_\_\_** **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_Игнатьев А.И.\_\_\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_Романов А.С.\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*2022 г.*

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой \_\_\_\_\_\_\_\_\_\_

(Индекс)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(И.О.Фамилия)

« \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 20 \_\_\_\_ г.

**ЗАДАНИЕ**

**на выполнение курсового проекта**

по дисциплине \_\_\_\_\_\_\_\_\_Распределенные системы обработки информации\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент группы ИУ7-21М

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Игнатьев Андрей Иванович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, имя, отчество)

Тема курсового проекта *Прототип интернет-магазина видеоигр*

Направленность КП (учебный, исследовательский, практический, производственный, др.)

учебный\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Источник тематики (кафедра, предприятие, НИР) кафедра\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

График выполнения проекта: 25% к 5 нед., 50% к 10 нед., 75% к 13 нед., 100% к 16 нед.

***Задание*** *Разработать систему, представляющую собой прототип интернет-магазина видеоигр. Система должна состоять из микросервисов, каждый из которых отвечает за свою задачу:* *сервис пользовательского интерфейса; сервис авторизации и аккаунтов; сервис игр; сервис транзакций; сервис статистики; сервис агрегирования запросов.* *Каждый сервис при необходимости имеет доступ к связанной с ним базе данных, но не имеет доступа к базам данных других сервисов. Запросы пользователей требуют авторизацию для всех случаев, кроме запросов регистрации, создания сессии и доступа к информации об играх. Все ошибки должны обрабатываться. В случае недоступности некритичного функционала должна осуществляться деградация функциональности. Все действия на сервисах должны логироваться. Все сервисы собираются и разворачиваются через CI/CD.*

***Оформление курсового проекта:***

Расчетно-пояснительная записка на 20-25 листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

*Слайды презентации.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Дата выдачи задания «27» марта 2022 г.

**Руководитель курсового проекта**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Романов А.С.

(Подпись, дата) (И.О.Фамилия)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_**Игнатьев А.И.

(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Оглавление

[РЕФЕРАТ 5](#_Toc103646468)

[Введение 6](#_Toc103646469)

[1. Аналитический раздел 7](#_Toc103646470)

[1.1. Описание системы 7](#_Toc103646471)

[1.2. Существующие аналоги 7](#_Toc103646472)

[1.2.1. Steam 7](#_Toc103646473)

[1.2.2. Epic Games Store (EGS) 7](#_Toc103646474)

[1.2.3. GOG 8](#_Toc103646475)

[1.2.4. Microsoft store 8](#_Toc103646476)

[1.3. Требования к системе с точки зрения пользователя 8](#_Toc103646477)

[1.4. Требования к программной реализации 9](#_Toc103646478)

[1.5. Состав системы 9](#_Toc103646479)

[1.6. Функциональные требования к сервисам 10](#_Toc103646480)

[1.6.1. Сервис пользовательского интерфейса и фронтенд 10](#_Toc103646481)

[1.6.2. Сервис авторизации и аккаунтов 11](#_Toc103646482)

[1.6.3. Сервис игр 11](#_Toc103646483)

[1.6.4. Сервис транзакций 11](#_Toc103646484)

[1.6.5. Сервис статистики 12](#_Toc103646485)

[1.6.6. Сервис агрегирования запросов 12](#_Toc103646486)

[2. Конструкторский раздел 14](#_Toc103646487)

[2.1. Ролевая модель 14](#_Toc103646488)

[2.2. Архитектура системы 14](#_Toc103646489)

[2.2.1. Сервис игр 15](#_Toc103646490)

[2.2.2. Сервис транзакций 16](#_Toc103646491)

[2.2.3. Сервис сессий 17](#_Toc103646492)

[2.2.4. Сервис статистики 19](#_Toc103646493)

[2.2.5. Сервис агрегации запросов (gateway) 20](#_Toc103646494)

[2.2.6. Сервис фронтенда 22](#_Toc103646495)

[2.3. Диаграмма последовательности действий 22](#_Toc103646496)

[3. Технологический раздел 23](#_Toc103646497)

[3.1. Схема базы данных 23](#_Toc103646498)

[3.2. Реализация фронтенда 23](#_Toc103646499)

[3.3. Реализация бекенда 24](#_Toc103646500)

[3.4. Сборка и развертывание системы 25](#_Toc103646501)

[3.5. Тестирование 25](#_Toc103646502)

[Заключение 26](#_Toc103646503)

[**Список литературы** 27](#_Toc103646504)

[Приложение А 28](#_Toc103646505)

РЕФЕРАТ

Расчетно-пояснительная записка 34 с., 3 рис., 10 табл., 10 источников, 1 прил.

Цель данной работы – разработать прототип магазина видеоигр с использованием микросервисной архитектуры.

Задачи работы:

* Проанализировать существующие аналоги;
* Сформулировать требования к системе и сервисам;
* Спроектировать систему и сервисы;
* Описать реализацию системы;
* Описать сборку, тестирование и развертывание системы.

В первой части работы приведены аналоги системы, сформулированы требования к системе и к сервисам. Во второй части работы приведена ролевая модель, описана архитектура системы, приведены спецификации сущностей системы, указаны методы API сервисов. В третьей части работы описаны особенности реализации сервисов, описаны сборка, тестирование и развертывание системы.

Введение

На данный момент все дольше контента переходит в цифровой формат. Хотя видеоигры на физических носителях все еще имеют аудиторию, все больше людей обращают внимание на цифровые копии. Объемы продаж цифровых копий на всех платформах заметно превосходят физические. Это подводит к мысли, что продажа видеоигр в интернет-магазине может быть прибыльнее, чем продажа физических копий. Данная работа ставит своей целью создание прототипа интернет-магазина видеоигр.

1. Аналитический раздел

В данном разделе представлено описание системы и ее компонентов, а также аналогов. Приведены требования к системе.

* 1. Описание системы

Система представляет собой прототип интернет-магазина видеоигр, реализованный в виде веб-приложения. Приложение должно позволять пользователю зарегистрироваться, войти в аккаунт и выйти из сессии. Пользователь должен иметь возможность просмотреть список игр и информацию о конкретной игре, а также купить игру. Купленные игры появляются у пользователя в библиотеке, где их можно просмотреть. Пользователь может также просмотреть информацию о себе, проверить и пополнить баланс счета магазина.

* 1. Существующие аналоги

Существует много разных магазинов видеоигр. Некоторые просто предоставляют возможность покупать игры, другие объединяют в себе функционал магазина, социальной сети и другое. К самым известным можно отнести такие площадки, как Steam, EGS, GOG и Microsoft store.

* + 1. Steam

Steam – площадка от компании Valve. Она объединяет в себе магазин, социальную сеть, форум, мастерскую и многие другие связанные с видеоиграми вещи. На данный момент аудитория площадки насчитывает миллионы пользователей по всему миру [1].

* + 1. Epic Games Store (EGS)

EGS – относительно недавно появившийся магазин от Epic Games, который позиционируется как основной конкурент Steam. EGS не предоставляет таких широких возможностей, как Steam, но в нем присутствуют временно эксклюзивные для платформы игры, а также частые раздачи бесплатных игр [2].

* + 1. GOG

Магазин от компании CD Project Red. Известен тем, что продает игры без антипиратской защиты, которая, как считается, может плохо влиять на работу игры, даже если она куплена легально [3].

* + 1. Microsoft store

Площадка от компании Microsoft, интегрированная в операционную систему Windows. Имеет общую библиотеку для платформ ПК и Xbox, а также возможность играть по подписке, не покупая сами игры. Помимо игр предоставляет также и различные приложения [4].

* 1. Требования к системе с точки зрения пользователя

Система должна предоставлять пользователям следующий функционал:

1. Регистрация пользователей в системе;
2. Авторизация пользователей посредством ввода логина и пароля;
3. Возможность выйти из сессии;
4. Разделение пользователей на 2 роли: пользователь и администратор;
5. Пользователи с ролью «пользователь» имеют доступ к следующему функционалу:
   1. Просмотр списка игр;
   2. Просмотр информации о конкретной игре;
   3. Покупка игры;
   4. Просмотр купленных игр (библиотеки);
   5. Просмотр информации о себе;
   6. Просмотр состояния счета магазина;
   7. Пополнение счета магазина;
6. Пользователи с ролью «администратор» имеют доступ к просмотру статистики работы системы.
   1. Требования к программной реализации
7. Система должна состоять из микросервисов, каждый из которых отвечает за свою область логики приложения;
8. Взаимодействие между сервисами осуществляется по протоколу HTTP;
9. Данные сервисов хранятся в базе данных. Каждый сервис имеет свою базу данных и не имеет доступа к базам данных других сервисов;
10. При недоступности систем портала должна осуществляться деградация функционала или выдача пользователю сообщения об ошибке;
11. Необходимо предусмотреть авторизацию пользователей через интерфейс веб-приложения;
12. Для запросов, обновляющих данные в нескольких узлах системы, нужно предусмотреть откат всей операции в случае возникновения ошибки.
    1. Состав системы

Система состоит из 6 сервисов:

1. Сервис пользовательского интерфейса;
2. Сервис авторизации и аккаунтов;
3. Сервис игр;
4. Сервис транзакций;
5. Сервис статистики;
6. Сервис агрегирования запросов (gateway сервис).
   1. Функциональные требования к сервисам

Здесь перечислены функциональные требования к каждому из сервисов.

* + 1. Сервис пользовательского интерфейса и фронтенд

Сервис пользовательского интерфейса представляет собой сервер, отдающий пользователю файлы веб-приложения. После получения и запуска веб-приложения пользователь работает с системой через него.

Веб-приложение (фронтенд) должно отвечать следующим требованиям:

1. Фронтенд принимает и обрабатывает команды пользователя, передаваемые через элементы интерфейса;
2. Фронтенд разрабатывается в формате толстого клиента и самостоятельно формирует веб-страницы на основе данных, полученных от gateway сервиса; рекомендуется использовать фреймворк Angular для разработки;
3. При необходимости получения данных от сервисов фронтенд отправляет на gateway сервис запросы по протоколу HTTP и получает ответ;
4. Обмен данными по протоколу HTTP осуществляется в формате JSON;
5. Для создания HTML-верстки целесообразно использовать компоненты из пакета Angular Material;
6. Фронтенд самостоятельно выполняет часть операций обработки данных, а именно их фильтрацию, сортировку и преобразования в пригодный для отображения в интерфейсе вид;
7. Фронтенд может кешировать данные, чтобы предотвратить лишние обращения к серверу.
   * 1. Сервис авторизации и аккаунтов

Сервис авторизации и аккаунтов хранит данные пользователей и сессий, а также осуществляет авторизацию и проверку токенов пользователей. Он должен удовлетворять следующим требованиям:

1. Сервис должен осуществлять взаимодействие с gateway сервисом по протоколу HTTP, передавать и принимать данные в формате JSON;
2. Сервис должен проводить авторизацию пользователей, управлять сессией пользователя;
3. Сервис должен отдавать информацию о пользователях;
4. Сервис использует JWT авторизацию с одним токеном, токен доступа имеет максимальное время жизни 30 минут;
5. Сервис должен давать возможность регистрации новых пользователей.
   * 1. Сервис игр

Сервис игр хранит данные об играх. Он должен удовлетворять следующим требованиям:

1. Сервис должен осуществлять взаимодействие с gateway сервисом по протоколу HTTP, передавать и принимать данные в формате JSON;
2. Сервис должен отдавать список всех игр и информацию о конкретной игре;
3. Список игр должен разделяться на страницы и подгружаться постепенно, номер и размер страницы задаются в запросе.
   * 1. Сервис транзакций

Сервис транзакций должен хранить информацию о покупках пользователями игр. Он должен удовлетворять следующим требованиям:

1. Сервис должен осуществлять взаимодействие с gateway сервисом по протоколу HTTP, передавать и принимать данные в формате JSON;
2. Сервис должен сохранять информацию о купленных пользователями играх, включая идентификаторы пользователя и игры и сумму покупки;
3. Сервис должен сохранять информацию о пополнениях баланса пользователей;
4. Сервис должен отдавать информацию о транзакциях пользователя.
   * 1. Сервис статистики

Сервис статистики собирает информацию о происходящих в системе событиях. Каждый сервис при совершении операции отправляет сообщение на этот сервис. Он должен удовлетворять следующим требованиям:

1. Сервис должен осуществлять взаимодействие с gateway сервисом по протоколу HTTP, передавать и принимать данные в формате JSON;
2. Сервис должен принимать записи о проводимых на сервисах операциях и сохранять их;
3. Сервис должен отдавать статистику о работе системы.
   * 1. Сервис агрегирования запросов

Данный сервис является единственным доступным для взаимодействия с фронтендом. Он выполняет роль прослойки между фронтендом и остальной системой. Он должен удовлетворять следующим требованиям:

1. Сервис должен осуществлять взаимодействие с фронтендом и другими сервисами по протоколу HTTP, принимать и передавать данные в формате JSON;
2. Сервис должен предоставлять методы API системы, каждый из которых (кроме регистрации, входа в систему и предоставления информации об играх) должен быть защищен JWT авторизацией;
3. Сервис должен разделять методы, доступные только пользователям и только администраторам;
4. Сервис должен реализовывать логику деградации функциональности при недоступности сервисов, необходимых для совершения операции;
5. Сервис должен выполнять откат сложных операций при возникновении в них ошибок;
6. Сервис должен получать данные от других сервисов и выполнять их предобработку, если операция этого требует.
7. Конструкторский раздел

В данном разделе описывается архитектура системы. Приведено описание ролевой модели и каждого компонента системы.

* 1. Ролевая модель

В системе присутствуют две роли пользователей:

* User – обычный пользователь магазина. Имеет счет, который может пополнять, а также может покупать игры;
* Admin – администратор. Может просматривать статистику по операциям, производимым сервисами бекенда. Не может покупать игры и пополнять счет.
  1. Архитектура системы

Рисунок 1 отображает архитектуру системы.

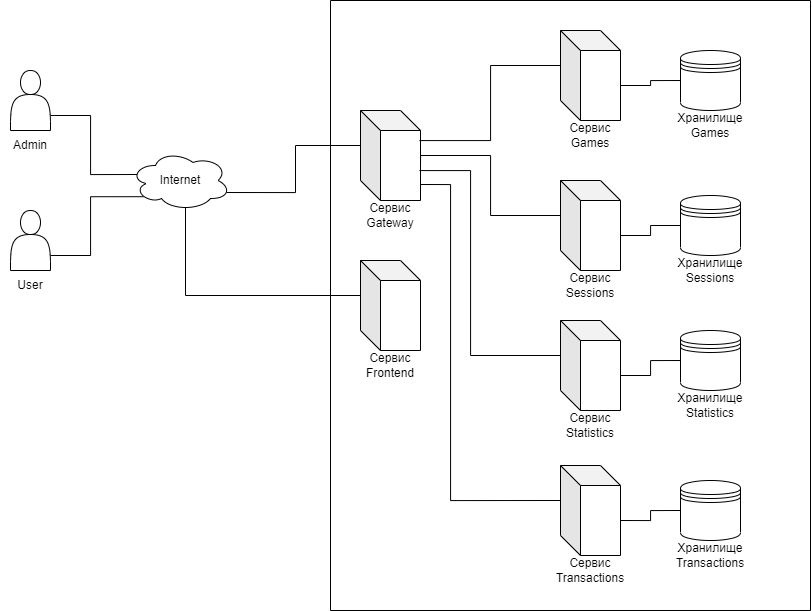


Рисунок 1. Архитектура системы

Сервисы работают на разных серверах. Базы данных для сервисов отдельные, сервисы имеют доступ только к ним и не имеют доступа к базам данных других сервисов. Пользователь имеет доступ к сервису фронтенда, с которого он получает веб-приложение. Веб-приложение связано с бэкендом через сервис gateway, который агрегирует запросы клиента и при необходимости связывается с остальными сервисами по протоколу HTTP. Все сервисы бэкенда имеют возможность отправлять статистику на сервис статистики. Для отправки статистики используется очередь, которая позволяет не потерять записи в случае недоступности сервиса статистики.

* + 1. Сервис игр

Сервис игр хранит информацию об играх. Сервис способен отдавать список игр с разделением на страницы, а также отдавать информацию о конкретной игре.

Методы API сервиса:

|  |  |
| --- | --- |
| URL | Описание |
| GET /games | Получить список игр.  Query параметры:   * page – номер страницы * size – размер страницы |
| GET /games/:id | Получить игру по ее id |

Сервис связан с базой данных, в которой присутствует таблица game, хранящая информацию об играх. Спецификация таблицы:

|  |  |  |
| --- | --- | --- |
| Столбец | Тип | Описание |
| id | int | Id игры, первичный ключ |
| name | varchar(64) | Название игры |
| developer | varchar(64) | Разработчик |
| publisher | varchar(64) | Издатель |
| price | float | Стоимость игры в рублях |
| description | varchar(4096) | Описание |
| requirements | varchar(2048) | Системные требования |

* + 1. Сервис транзакций

Сервис транзакций хранит информацию о транзакциях пользователей – покупках игр и пополнениях баланса, с возможностью получения этой информации для конкретного пользователя. Присутствует также возможность удалить транзакцию по ее id при необходимости отката.

Методы API сервиса:

|  |  |
| --- | --- |
| URL | Описание |
| GET /transactions | Получить все транзакции пользователя  Заголовки:   * X-User-Id – id пользователя |
| POST /transactions | Добавить транзакцию  Заголовки:   * X-User-Id – id пользователя |
| DELETE /transactions/:id | Удалить транзакцию по id |

Сервис связан с базой данных, в которой присутствует таблица transaction, хранящая информацию о транзакциях. Спецификация таблицы:

|  |  |  |
| --- | --- | --- |
| Столбец | Тип | Описание |
| id | int | Id транзакции, первичный ключ |
| userId | int | Id пользователя |
| gameId | int | Id купленной игры, может отсутствовать |
| type | enum | Тип транзакции, возможные значения: fill, buy |
| sum | float | Сумма транзакции в рублях |

* + 1. Сервис сессий

Сервис сессий хранит информацию о пользователях и сессиях, позволяет создавать и редактировать пользователей, создавать и удалять сессии, проводить валидацию jwt токенов. Jwt токены генерируются при создании сессии. Пароли пользователей хранятся в хешированном виде с использованием алгоритма md5 [5].

Методы API сервиса:

|  |  |
| --- | --- |
| URL | Описание |
| POST /auth/session | Создать сессию |
| DELETE /auth/session/:id | Удалить сессию по id |
| POST /auth/validate | Провалидировать jwt токен |
| GET /auth/session | Получить сессию по токену  Заголовки:   * Authorization – токен |
| GET /users | Получить пользователя по id  Заголовки:   * X-User-Id – id пользователя |
| GET /users/name | Получить пользователя по имени  Заголовки:   * X-User-Name – имя пользователя |
| POST /users | Создать пользователя |
| PATCH /users | Отредактировать пользователя |

Сервис связан с базой данных, в которой присутствуют таблицы session с информацией о сессиях и user с информациях о пользователях. Спецификация таблицы session:

|  |  |  |
| --- | --- | --- |
| Столбец | Тип | Описание |
| id | int | Id сессии, первичный ключ |
| userId | int | Id пользователя сессии |
| token | varchar(256) | Jwt токен сессии |
| createDate | timestamp with time zone | Время создания сессии |

Спецификация таблицы user:

|  |  |  |
| --- | --- | --- |
| Столбец | Тип | Описание |
| id | int | Id пользователя, первичный ключ |
| email | varchar(128) | email пользователя |
| username | varchar(64) | Имя пользователя, уникальное |
| password | varchar(256) | Пароль, хешированный алгоритмом md5 |
| role | varchar(16) | Роль пользователя (User или Admin) |
| balance | float | Баланс пользователя в рублях |

* + 1. Сервис статистики

Сервис статистики хранит записи о происходящих в системе действиях, которые он получает от других сервисов бекенда. Он предоставляет возможность добавить запись и получить все записи для конкретного сервиса в диапазоне дат.

Методы API сервиса:

|  |  |
| --- | --- |
| URL | Описание |
| GET /statistics | Получить все записи о действиях для указанного сервиса в указанном диапазоне дат  Query параметры:   * service – название сервиса * dateFrom – начальная дата * dateTo – конечная дата |
| POST /statistics | Добавить запись статистики |

Сервис связан с базой данных, в которой присутствует таблица statistic, хранящая информацию о действиях сервисов. Спецификация таблицы:

|  |  |  |
| --- | --- | --- |
| Столбец | Тип | Описание |
| id | int | Id записи, первичный ключ |
| service | varchar(32) | Название сервиса |
| description | varchar(512) | Описание записи |
| timestamp | Timestamp with time zone | Дата создания записи |

* + 1. Сервис агрегации запросов (gateway)

Gateway сервис предоставляет внешнее API системы и выступает в роли агрегатора запросов. Сервис принимает запросы от клиентского приложения по протоколу HTTP и отправляет ответы на них. При необходимости сервис осуществляет взаимодействие с другими сервисами бекенда. Сервис осуществляет валидацию данных запросов и инициирует проверки авторизации и прав доступа. Проверка авторизации осуществляется для всех запросов, кроме запросов регистрации и авторизации пользователя, и запросов на получение информации об играх.

Методы API сервиса:

|  |  |
| --- | --- |
| URL | Описание |
| POST /api/auth/login | Создание сессии пользователя |
| DELETE /api/auth/logout | Выход пользователя из сессии. Требуется авторизация |
| POST /api/auth/register | Регистрация нового пользователя |
| GET /api/games | Получить список игр.  Query параметры:   * page – номер страницы * size – размер страницы |
| GET /api/games/:id | Получить игру по ее id |
| POST /api/games/:id/buy | Купить игру с заданным id. Требует авторизации. Доступно только пользователям с ролью User |
| GET /api/statistics | Получить записи статистики по заданным параметрам. Требует авторизации. Доступно только пользователям с ролью Admin  Query параметры:   * service – название сервиса * dateFrom – начальная дата * dateTo – конечная дата |
| GET /api/me | Получить информацию о пользователе. Требует авторизации. Доступно только пользователям с ролью User |
| GET /api/me/library | Получить список купленных пользователем игр. Требует авторизации. Доступно только пользователям с ролью User |
| GET /api/me/wallet | Получить баланс счета пользователя. Требует авторизации. Доступно только пользователям с ролью User |
| PATCH /api/me/wallet | Пополнить баланс пользователя. Требует авторизации. Доступно только пользователям с ролью User |

Все операции, требующие авторизации, предполагают наличие заголовка Authorization со значением формата “Bearer <jwt-token>”, по которому проверяется факт авторизации и из которого выводится производящий операцию пользователь. Описание API gateway сервиса в формате openAPI приведено в приложении А.

Сервис предусматривает действия при недоступности требуемых для успешного выполнения операции сервисов. Для гарантии записи статистики и отката транзакций используются очереди задач. Для обеспечения деградации функциональности при недоступности некритичных источников данных используется подход “Circuit breaker” [6].

* + 1. Сервис фронтенда

Сервис фронтенда хранит статические файлы клиентского приложения и отдает их пользователю при использовании сайта. Веб-приложение имеет формат SPA и само рендерит страницы, поэтому этот сервис только отдает статические файлы приложения.

* 1. Диаграмма последовательности действий

Рисунок 2 демонстрирует диаграмму последовательности действий на примере операции покупки игры. На диаграмме показано, как взаимодействуют сервисы.

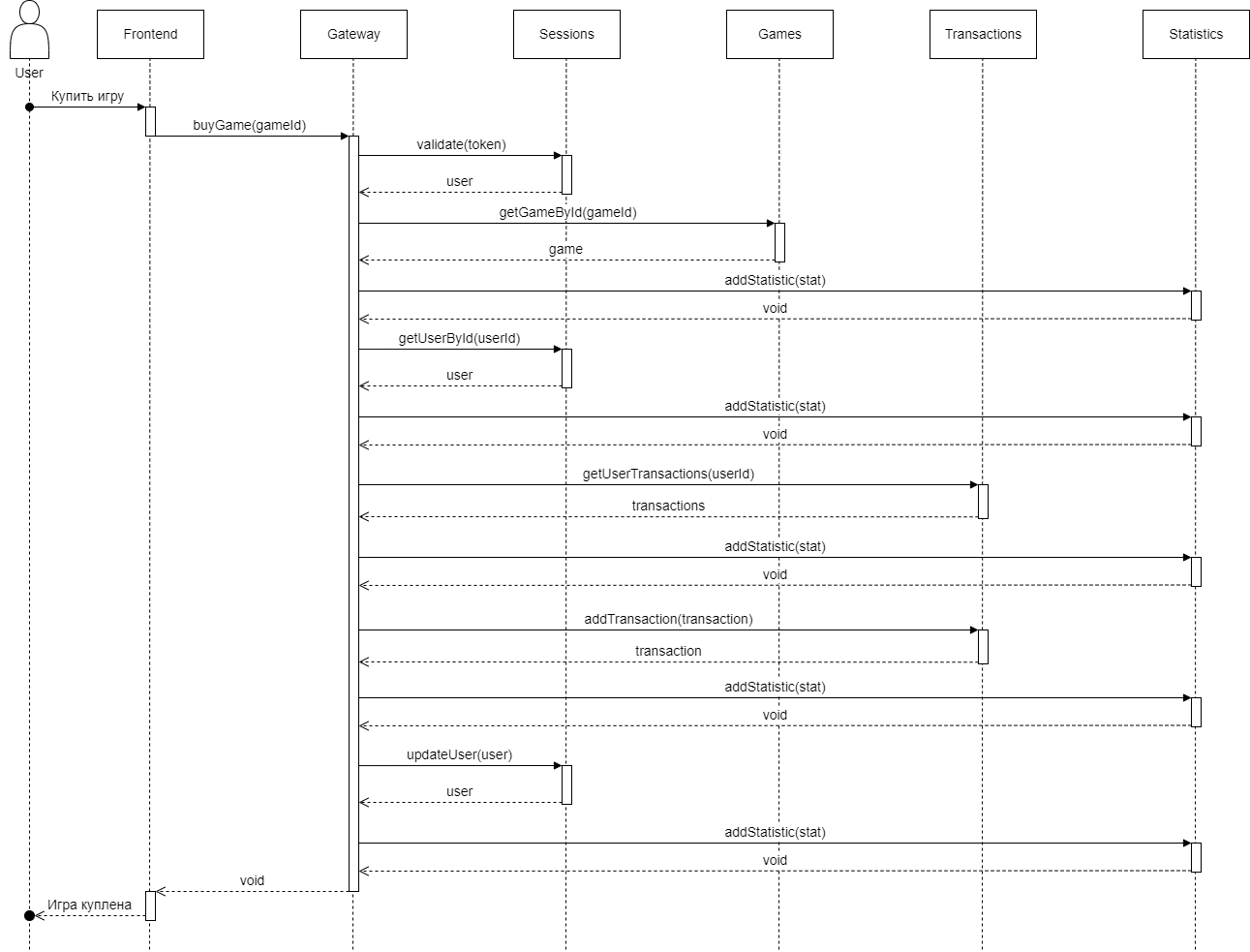


Рисунок 2. Диаграмма последовательности действий для операции покупки игры

1. Технологический раздел

В этом разделе приведены описания структур данных и описана реализация системы.

* 1. Схема базы данных

Рисунок 3 показывает общую схему базы данных системы. Каждый сервис имеет свою базу данных, поэтому отношения между большинством таблиц фактически не контролируются на уровне СУБД. Спецификации таблиц приведены в разделах 2.2.1. – 2.2.4..

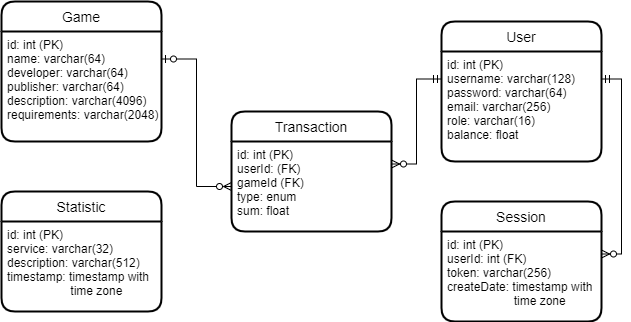


Рисунок 3. Схема базы данных

* 1. Реализация фронтенда

Фронтенд приложение реализовано с использованием фрейморка Angular [7]. Приложение выполнено по принципу SPA, то есть страница сайта загружается только один раз, после чего перерисовывается в соответствии с действиями пользователя. Интерфейс приложения сверстан с использованием компонентов Angular Material [8], а также собственной CSS разметки.

Приложение имеет экраны, доступные только авторизованным пользователям, только не авторизованным пользователям и всем пользователям. Экраны списка игр и информации об игре доступны всем пользователям. Экраны авторизации и регистрации доступны только неавторизованным пользователям. Экран информации о пользователе доступен только авторизованным пользователям и ролью User. Экран со статистикой работы системы доступен только авторизованным пользователям с ролью Admin.

* 1. Реализация бекенда

Все сервисы бекенда реализованы с использованием фреймворка nest.js [9]. Взаимодействие между сервисами осуществляется по протоколу HTTP. В качестве СУБД используется postgresql, взаимодействие с ней осуществляется посредством typeorm. Сервисы сессий, игр, транзакций и статистики имеют связь с базами данных. Каждый из этих сервисов имеет доступ только к своей базе данных.

Проверка авторизации инициируется gateway сервисом и осуществляется по его запросам на сервисе сессий. Вся работа с паролями и jwt токенами осуществляется также на сервисе сессий. Остальные сервисы авторизацию не проводят. К общедоступным (не защищенным авторизацией) методам относятся методы регистрации, авторизации, получения списка игр и информации об игре по ее id. Остальные методы требуют наличия заголовка авторизации.

Сервис gateway осуществляет валидацию данных входящих запросов и в случае неудачи возвращает ошибку. Остальные сервисы валидацию не проводят.

Сервис gateway имеет средства обеспечения функциональности системы в случае отказа ее узлов. Если недоступен источник некритичных данных, используется значение-заглушка. В случае недоступности критичных для операции сервисов клиенту возвращается ошибка 503 и в случае операций с транзакциями выполняется их откат.

* 1. Сборка и развертывание системы

Процесс сборки и развертывания системы осуществляется с помощью CI/CD. После прохождения этапов сборки и тестирования сервисы приложения собираются в отдельные докер-контейнеры. Собранные контейнеры публикуются на Docker Hub [10].

Репозиторий с проектом содержит также файл docker compose, позволяющий развернуть систему локально. Предварительно каждый сервис требуется собрать с помощью команды npm run build.

* 1. Тестирование

Тестирование системы производилось методом черного ящика. Тестировались сервисы как по отдельности, так и вместе. Прорабатывались различные сценарии, в том числе:

* Регистрация пользователя;
* Авторизация пользователя;
* Просмотр списка игр и информации о конкретной игре;
* Покупка игры;
* Просмотр информации о пользователе;
* Просмотр статистики;
* Выход из аккаунта.

Для тестирования отказоустойчивости отключались некоторые сервисы и проверялось поведение системы.

В CI/CD также присутствует этап автоматического тестирования. Он проверяет систему на отсутствие ошибок при инстанцировании классов. Более подробное автоматическое тестирование было решено не проводить ввиду отсутствия временных ресурсов на разработку тестов.

Заключение

По результатам работы была спроектирован и разработан прототип магазина видеоигр. Система состоит из 6 сервисов: фронтенда, сервиса агрегации запросов, сервиса игр, сервиса сессий, сервиса транзакций, сервиса статистики.

Сервисы игр, сессий, транзакций, статистики имеют доступ к связанным с ними базам данных, но не имеют доступа к базам данных других сервисов. Запросы пользователей требуют авторизацию для всех случаев, кроме запросов регистрации, создания сессии и доступа к информации об играх, а все ошибки обрабатываются. В случае недоступности некритичного функционала осуществляется деградация функциональности. Все действия на сервисах логируются на сервисе статистики. Все сервисы собираются и разворачиваются через CI/CD.

# Список литературы

|  |  |
| --- | --- |
| [1] | Valve, «Steam» [В Интернете]. URL: https://store.steampowered.com/about/. [Дата обращения: 07.04.2022]. |
| [2] | Epic Games, «Epic Games Store» [В Интернете]. URL: https://store.epicgames.com/ru/. [Дата обращения: 07.04.2022]. |
| [3] | GOG.COM, «GOG.com» [В Интернете]. URL: https://www.gog.com/ru/about\_gog. [Дата обращения: 07.04.2022]. |
| [4] | Microsoft, «Приложения для Windows - Microsoft Store» [В Интернете]. URL: https://www.microsoft.com/ru-RU/store/apps. [Дата обращения: 07.04.2022]. |
| [5] | А. Малахов, «Хеширование и расшифровка MD5 хеш-кода» [В Интернете]. URL: https://www.internet-technologies.ru/articles/newbie/heshirovanie-i-rasshifrovka-md5-hesh-koda.html. [Дата обращения: 10.05.2022]. |
| [6] | Microsoft, «Circuit Breaker pattern» [В Интернете]. URL: https://docs.microsoft.com/en-us/azure/architecture/patterns/circuit-breaker. [Дата обращения: 10.05.2022]. |
| [7] | Google, «Angular» [В Интернете]. URL: https://angular.io/. [Дата обращения: 10.05.2022]. |
| [8] | Google, «Angular Material UI component library» [В Интернете]. URL: https://material.angular.io/. [Дата обращения: 10.05.2022]. |
| [9] | «NestJS - A progressive Node.js framework» [В Интернете]. URL: https://nestjs.com/. [Дата обращения: 10.05.2022]. |
| [10] | «Docker Hub» [В Интернете]. URL: https://hub.docker.com/. [Дата обращения: 10.05.2022]. |

Приложение А

openapi: 3.0.3

info:

title: Gateway

description: Gateway

version: 1.0.0

tags:

- name: Games

- name: Auth

- name: User

- name: Stats

paths:

/api/auth/register:

post:

tags:

- Auth

summary: Регистрация пользователя

requestBody:

required: true

content:

application/json:

schema:

$ref: '#/components/schemas/RegisterRequest'

responses:

'200':

description: Регистрация успешна

'409':

description: Пользователь уже существует

content:

application/json:

schema:

$ref: '#/components/schemas/ConflictException'

/api/auth/login:

post:

tags:

- Auth

summary: Логин пользователя

requestBody:

required: true

content:

application/json:

schema:

$ref: '#/components/schemas/UserLogin'

responses:

'200':

description: Сессия создана

content:

application/json:

schema:

$ref: '#/components/schemas/Session'

'401':

description: Ошибка авторизации

content:

application/json:

schema:

$ref: '#/components/schemas/UnauthorizedException'

/api/auth/logout:

delete:

tags:

- Auth

summary: Логаут пользователя

parameters:

- in: header

name: Authorization

required: true

schema:

type: string

description: JWT token

example: 'Bearer 12d4f234rvw21g34'

responses:

'200':

description: Логаут успешен

'401':

description: Токен недействителен

content:

application/json:

schema:

$ref: '#/components/schemas/UnauthorizedException'

/api/games:

get:

tags:

- Games

summary: Получить список игр

parameters:

- in: query

name: page

schema:

type: number

minimum: 0

- in: query

name: size

schema:

type: number

minimum: 1

responses:

'200':

description: Игры загружены

content:

application/json:

schema:

type: array

items:

$ref: '#/components/schemas/Game'

'400':

description: Некорректный запрос

content:

application/json:

schema:

$ref: '#/components/schemas/BadRequestException'

/api/games/{id}:

get:

tags:

- Games

summary: Получить игру по id

parameters:

- in: path

name: id

required: true

schema:

type: number

responses:

'200':

description: Игра загружена

content:

application/json:

schema:

$ref: '#/components/schemas/Game'

'404':

description: Игра не найдена

content:

application/json:

schema:

$ref: '#/components/schemas/NotFoundException'

/api/games/{id}/buy:

post:

tags:

- Games

summary: Купить игру

parameters:

- in: header

name: Authorization

required: true

schema:

type: string

description: JWT token

example: 'Bearer 12d4f234rvw21g34'

- in: path

name: id

required: true

schema:

type: number

requestBody:

content:

application/json:

schema:

type: object

responses:

'200':

description: Игра куплена

'404':

description: Игра не найдена

content:

application/json:

schema:

$ref: '#/components/schemas/NotFoundException'

'401':

description: Токен недействителен

content:

application/json:

schema:

$ref: '#/components/schemas/UnauthorizedException'

'422':

description: Недостаточно средств

content:

application/json:

schema:

$ref: '#/components/schemas/UnprocessableEntityException'

'409':

description: Игра уже куплена

content:

application/json:

schema:

$ref: '#/components/schemas/ConflictException'

'503':

description: Сервис недоступен

content:

application/json:

schema:

$ref: '#/components/schemas/ServiceUnavailableException'

'403':

description: Пользователь не найден

content:

application/json:

schema:

$ref: '#/components/schemas/ForbiddenException'

/api/me:

get:

tags:

- User

summary: Получить информацию о пользователе

parameters:

- in: header

name: Authorization

required: true

schema:

type: string

description: JWT token

example: 'Bearer 12d4f234rvw21g34'

responses:

'200':

description: Краткая информация о пользователе

content:

application/json:

schema:

$ref: '#/components/schemas/User'

'401':

description: Токен недействителен

content:

application/json:

schema:

$ref: '#/components/schemas/UnauthorizedException'

'503':

description: Сервис недоступен

content:

application/json:

schema:

$ref: '#/components/schemas/ServiceUnavailableException'

'404':

description: Игра не найдена

content:

application/json:

schema:

$ref: '#/components/schemas/NotFoundException'

/api/me/library:

get:

tags:

- User

summary: Получить купленные пользователем игры

parameters:

- in: header

name: Authorization

required: true

schema:

type: string

description: JWT token

example: 'Bearer 12d4f234rvw21g34'

responses:

'200':

description: Игры пользователя

content:

application/json:

schema:

type: array

items:

$ref: '#/components/schemas/Game'

'401':

description: Токен недействителен

content:

application/json:

schema:

$ref: '#/components/schemas/UnauthorizedException'

'503':

description: Сервис недоступен

content:

application/json:

schema:

$ref: '#/components/schemas/ServiceUnavailableException'

'404':

description: Игра не найдена

content:

application/json:

schema:

$ref: '#/components/schemas/NotFoundException'

/api/me/wallet:

get:

tags:

- User

summary: Получить баланс пользователя

parameters:

- in: header

name: Authorization

required: true

schema:

type: string

description: JWT token

example: 'Bearer 12d4f234rvw21g34'

responses:

'200':

description: Баланс пользователя

content:

application/json:

schema:

$ref: '#/components/schemas/FillBalanceRequest'

'401':

description: Токен недействителен

content:

application/json:

schema:

$ref: '#/components/schemas/UnauthorizedException'

'503':

description: Сервис недоступен

content:

application/json:

schema:

$ref: '#/components/schemas/ServiceUnavailableException'

'404':

description: Игра не найдена

content:

application/json:

schema:

$ref: '#/components/schemas/NotFoundException'

patch:

tags:

- User

summary: Пополнить баланс пользователя

parameters:

- in: header

name: Authorization

required: true

schema:

type: string

description: JWT token

example: 'Bearer 12d4f234rvw21g34'

requestBody:

required: true

content:

application/json:

schema:

$ref: '#/components/schemas/FillBalanceRequest'

responses:

'200':

description: Баланс пользователя обновлен

content:

application/json:

schema:

$ref: '#/components/schemas/FillBalanceRequest'

'401':

description: Токен недействителен

content:

application/json:

schema:

$ref: '#/components/schemas/UnauthorizedException'

'503':

description: Сервис недоступен

content:

application/json:

schema:

$ref: '#/components/schemas/ServiceUnavailableException'

'404':

description: Игра не найдена

content:

application/json:

schema:

$ref: '#/components/schemas/NotFoundException'

'400':

description: Некорректная сумма

content:

application/json:

schema:

$ref: '#/components/schemas/BadRequestException'

/api/statistics:

get:

tags:

- Stats

summary: Получить статистику работы системы для сервиса

parameters:

- in: header

name: Authorization

required: true

schema:

type: string

description: JWT token

example: 'Bearer 12d4f234rvw21g34'

- in: query

name: service

required: true

schema:

type: string

- in: query

name: dateFrom

schema:

type: string

format: date

- in: query

name: dateTo

schema:

type: string

format: date

responses:

'200':

description: Статистика

content:

application/json:

schema:

type: array

items:

$ref: '#/components/schemas/Statistic'

'401':

description: Токен недействителен

content:

application/json:

schema:

$ref: '#/components/schemas/UnauthorizedException'

'400':

description: Некорректный запрос

content:

application/json:

schema:

$ref: '#/components/schemas/BadRequestException'

components:

schemas:

RegisterRequest:

type: object

properties:

login:

type: string

password:

type: string

email:

type: string

format: email

UserLogin:

type: object

properties:

username:

type: string

password:

type: string

FillBalanceRequest:

type: object

properties:

amount:

type: number

minimum: 1

Statistic:

type: object

properties:

service:

type: string

message:

type: string

type:

type: string

enum:

- Action

- Error

timestamp:

type: string

format: date-time

User:

type: object

properties:

id:

type: number

email:

type: string

username:

type: string

Session:

type: object

properties:

id:

type: number

userId:

type: number

token:

type: string

createDate:

type: string

Game:

type: object

properties:

id:

type: integer

name:

type: string

developer:

type: string

publisher:

type: string

price:

type: number

description:

type: string

requirements:

type: string

# Nest.JS builtins

ConflictException:

type: object

UnauthorizedException:

type: object

BadRequestException:

type: object

NotFoundException:

type: object

UnprocessableEntityException:

type: object

ForbiddenException:

type: object

ServiceUnavailableException:

type: object